ADC-100 Manual v1.2 http://www.programmation.fr

1. Introduction
2. Connecting to PC
3. Technical Specifications

Writing you own software overview

C

Turbo Pascal

C/C++ Win

Delphi

Visual Basic

Excel

HP Vee

Linux

Other Information

Contact Pico

Technical Support

Signal Conditioners

Safety

Legal

Print This Manual

Safety Warning

We strongly recommend that you read the general safety information below before using your product for the first time. If the equipment is not used in the manner specified, then the protection provided may be impaired. This could result in damage to your computer and/or injury to yourself or others.

Maximum input range
The ADC-100 is designed to measure voltages up to the range -20V to +20V. Any voltages in excess of ±100V may cause permanent damage to the unit.

Mains Voltages
No Pico products are designed for use with mains voltages. To measure mains we recommend the use of a differential isolating probe specifically designed for such measurements.

Safety Grounding
The ground of every product is connected directly to the ground of your computer via the provided interconnecting cable. This is done in order to minimise interference. Always use the provided cable to attach the product to your computer.

As with most oscilloscopes and data loggers, you should take care to avoid connecting the ground input of the product to anything which may be at some voltage other than ground. If in doubt, use a meter to check that there is no significant AC or DC voltage. Failure to check may cause damage to the product and/or computer and could cause injury to yourself or others.

You should assume that the product does not have a protective safety earth. Misconfiguration and/or use on voltages outside the maximum input range can be hazardous.

Repairs
The unit contains no user serviceable parts: repair or calibration of the unit requires specialised test equipment and must be performed by Pico Technology Limited or their authorised distributors.

Legal info

Grant of license
The material contained in this release is licensed, not sold. Pico Technology Limited grants a license to the person who installs this software, subject to the conditions listed below.
Access
The licensee agrees to allow access to this software only to persons who have been informed of and agree to abide by these conditions.
Usage
The software in this release is for use only with Pico products or with data collected using Pico products.
Copyright
Pico Technology Limited claims the copyright of and retains the rights to all material (software, documents etc) contained in this release.
The user may copy and distribute the entire release in installable form, but you may not copy individual items within the release other than for backup purposes.
Liability
Pico Technology and its agents shall not be liable for any loss, damage or injury, howsoever caused, related to the use of Pico Technology equipment or software, unless excluded by statute.
Fitness for purpose
No two applications are the same, so Pico Technology cannot guarantee that its equipment or software is suitable for a given application. It is therefore the user's responsibility to ensure that the product is suitable for the user's application.
Mission critical applications
Because the software runs on a computer that may be running other software products, and may be subject to interference from these other products, this license specifically excludes usage in 'mission critical' applications, for example life support systems.
Viruses
This software was continuously monitored for viruses during production, however the user is responsible for virus checking the software once it is installed.
Support
If you are unsatisfied with the performance of this software, please contact our technical support staff. If we believe that there is a problem, we will try to fix the problem within a reasonable timescale. If you are still unsatisfied, please return the product and software to your supplier within 28 days of purchase for a full refund.
Upgrades
Pico Technology provides upgrades, free of charge, from our web site www.picotech.com. We reserve the right to charge for updates or replacements sent out on physical media.

Trade marks
Windows is a registered trademark of Microsoft Corporation.

Pico Technology Limited, DrDAQ and PicoScope are internationally registered trade marks.

Introduction

The PICO ADC-100 and ADC-101 are medium speed analog to digital converters with two analog input channels and programmable input voltage ranges. They can be used as a virtual instrument (oscilloscope, spectrum analyser and meter) with the PicoScope program, or as a data logger using PicoLog. Alternatively, you can use the ADC-100 driver software to develop your own programs to collect and analyse data from the unit.

This manual describes the physical and electrical properties of the ADC-100 and ADC-101, and explains how to use the Windows software drivers. For information about the software supplied with the unit, please refer to the following documents:

PicoLog for Windows	Online help file
PicoScope for Windows	Online help file
DOS drivers	ADC100.TXT file
PicoLog for DOS	PL.TXT file

Connecting to the PC

The ADC-100 and ADC-101 can be connected to the PC in two ways:

- directly to a printer port on the computer
- to a USB port on the computer, via a PICO USB parallel port adapter.

Printer port operation

When you install the application software from the PICO CD, the computer will ask you which port to use. You should select LPT1, LPT2 or LPT3.

To use the ADC, you should connect it to the printer port on your computer, either directly or using a good quality extension cable.

USB port operation

Please note that USB printer port interfaces are not suitable for use with Pico products. If you wish to connect a Pico product to a USB port, you will need a Pico USB Parallel Port adapter. You will also need Windows 98, ME, 2000 or XP.

When you install the application software from the PICO CD:

- when the computer asks you which port to use, you should select USB-PP1

Once the USB driver software is installed, connect the Pico USB parallel port adapter to your PC: the computer will automatically configure the drivers.

See Streaming for more information about the advanages of operating via a Pico USB parallel port.

Checking the installation

To check that the unit is working, start up the PicoScope program and then connect a voltage source to the BNC connector. The ADC has the same connectors as an oscilloscope, so you can use standard oscilloscope probes.

PicoScope should now display the voltage that you have connected. If you are using scope probes, when you touch the scope probe tip with your finger, you should see a small 50Hz mains signal on the screen.

If you have connected the ADC to a printer port other than the port specified when you installed the software, you will need to go to the Setup panel and then change the port number to the appropriate value (USB port numbers begin with USB-PPx. If you have more than one USB parallel port, they will be numbered according to the order they are plugged into the PC). You will need to exit and re-enter the software to activate the change.

Technical Specifications

	ADC-100	**ADC-101**
Resolution	12 bits	
Number of input channels	2	
Typical sampling rate (measured on 100MHz Pentium)	Windows 3.1/95/98: 100kS/s Windows NT: 40kS/s	
Linearity	±4 LSB at 25°C	
Accuracy	±2%	
Input overvoltage protection	±100V	
Input type	single ended	differential
Input impedance	1MΩ	
Spectrum Range	0-50kHz	
Dynamic Range	70dB	
Input voltage ranges	±50mV to ±20V	±100mV to ±100V
AC/DC switch	Manual	None
Input connector	BNC	
Output connector	25 way male D-type (connects to PC printer port)	

asynchronously

The Pico USB parallel port takes samples from the product, under it's own control, and stores them until the computer has time to read them. The term asynchronous is used to indicate that the time the samples are collected is not the same as the time the PC requests them.

Streaming

If a device is connected to a PICO USB parallel port, data is collected asynchronously, without any intervention from the PC. This gives considerably more reliable data collection, and sampling does not interfere with the operation of the your computer.

When collecting data from a streaming device using the drivers, three modes are available:

BM_SINGLE - collect a single block of data and exit
BM_WINDOW - collect a series of overlapping blocks of data
BM_STREAM - collect a continuous stream of data.

BM_SINGLE is useful when you wish to collect data at high speed for a relatively short period. For example, to collect 1000 readings in 50ms.

BM_WINDOW is useful when collecting several blocks of data at relatively low speeds- for example when collecting 10000 samples over 10 seconds. Collecting a sequence of SINGLE blocks like this would take 10 seconds for each block, so displayed data would not be updated frequently. Using windowing, it is possible to ask for a new block more frequently, for example every second, and to receive a block containing nine seconds of data that have already been seen and one second of new data. The block is effectively a 10-second ‘window’ that advances one second each time.

BM_STREAM is useful when you need to collect data continuously for long periods. In principle, it would be possible to collect data indefinitely. Each time adc100_get_values is called, it returns the new readings since the last time it was called. No_of_values passed to adc100_run must be sufficient to ensure that the buffer does not overflow between successive calls to adc100_get_values. For example, if you call adc100_get_values every second, and you are collecting 500 samples per second, no_of_values must be at least 500, and preferably 1000 to give some allowance for delays in the operating system.

DOS stand-alone program

The program ADC100.exe reads in a block of data values from the ADC-100 and writes them to a file, as raw ADC values. See scaling for information on converting the readings to volts.

To run the program, type in

```
adc100 [-options] filename
```

where options are as listed below and filename is the name of the file to write the data to.

```
adc100 @control
```

where control is the name of a control file containing a number of options, one per line.

The following table defines the options:

Option	Example	Meaning
-p	...-p1	Get data from LPT1
-m	-ma	get data from channel A only
	-mb	get data from channel B only
	-mab	get data from both channels
-I	..-I10	wait 10us between each reading
-n	..-n1000	read in 1000 values from one/both channels
-t	-tn	no trigger
	-tm	manual trigger (press a key to start)
	-tra1000	trigger on channel A rising above 1000
	-tfb1000	trigger on channel B falling below 1000
-r	-ra20000	set channel A range to +/-20V
	-rb100	set channel B range to ñ100mV
-b	-b	binary file (ie not text)

Drivers

The ADC-100 and ADC-101 are supplied with driver routines that you can build into your own programs. There is also a DOS stand alone program that can be used for collecting large blocks of data at high speed.

Once you have installed the software, the DRIVERS sub-directory contains the drivers and a selection of examples of how to use the drivers. It also contains a copy of this help file in text format.

The driver routine is supplied as object files for DOS, and as Dynamic Link Libraries for Windows 3.1, 95/98/ME and NT/200/XP

Note that there are a number of differences between the DOS driver and the Windows drivers. See adc100.txt for information about the DOS driver.

The Windows DLLs can be used with any programming language or application that can interface with DLLs- for example, C, Delphi, Visual Basic, Excel, Labview, etc. The DRIVERS directory contains example programs for several popular programming languages or applications: some of these examples are fairly simple, but the C console mode example, a100con.c, shows how to use all facilities in the driver.

The driver is capable of supporting up to three units connected to printer ports (one each on LPT1, LPT2 and LPT3) and up to four PICO USB parallel port units. The units can be any mixture of ADC-100 and ADC-101.

The following table specifies the function of each of the routines in the Windows drivers:

Routine	**Function**
adc100_get_driver_version	Check that this is the correct driver
adc100_open_unit	Open the driver to use a specified parallel port
adc100_set_unit	Select which ADC-100 unit to use
adc100_close_unit	Close the specified port
adc100_set_range	Set the input voltage range
adc100_get_value	Get a single reading from one channel
adc100_is_streaming	Check whether the device supports streaming (USB only)
adc100_run	Start the unit recording
adc100_ready	Check whether the data recording is completed
adc100_stop	Abort data collection
adc100_set_trigger	Set a trigger event from a specified channel
adc100_set_interval	Set the channels and time interval for the next call to adc100_get_values, or adc100_get_times_and_values
adc100_get_values	Get a block of readings at fixed intervals
adc100_get_times_and_values	Get a block of readings and their times, at fixed intervals
adc100_get_unit_info	Get information about an ADC100 unit

The driver offers the following facilities:

specify the printer port that is connected to the ADC-100
take a single reading from a specified channel
specify a trigger event from a specified channel (only available in block mode)
collect a block of samples at fixed time intervals from one or more channels

You can specify a sampling interval from 10us to a second. If you specify an interval that is shorter than your computer can manage, the driver will tell you how long it will actually take to collect the specified number of samples. After allowing for this, the timing accuracy under DOS is better than 1% for a block of 1000 samples at all sampling rates.

Under Windows, if you connect the product to the computer via a PICO USB parallel port, timing is completely reliable. However, if you connect the product to the computer via the printer port, the sampling may be affected by Windows activities. At the least, there will be gaps in the data every 55 milliseconds due to the Windows timer function. There will be additional gaps if you move the mouse, or have other programs running. We therefore recommend using the adc100_get_values_and_times routine, so that you can determine the exact time that each reading was taken.

The normal calling sequence to collect a block of data is as follows:

Check that the driver version is correct
Open the driver
Set trigger mode (if required)
Set sampling mode (channels and time per sample)

While you want to take measurements,
    Run
    While not ready
        Wait
    End while
    Get a block of data
End While
Close the driver

Scaling

The ADC-100 and ADC-101 are 12-bit analog to digital converters. This means that they produce values in the range 0 to 4095 to represent the currently selected input voltage range. To convert from ADC readings to Volts, you should subtract half of the 2048, multiply by the currently selected voltage range and divide by 2048. Thus, on the 5V range, an ADC reading of 3135 represents (3135-2048) x 5 / 2048 = 2.654 Volts.

adc100_get_driver_version

```
PREF1 short PREF2 adc100_get_driver_version (void);
```

This routine returns the version number of the ADC100/101 driver. You can use it to check that your application is used only with the driver version that it was designed for use with.

Generally speaking, new driver versions will be fully backward compatible with earlier versions, though the converse is not always true, so it should be safe to check that the driver version is greater than or equal to the version that it was designed for use with.

The version is a two-byte value, of which the upper byte is the major version and the lower byte is the minor version.

adc100_open_unit

```
PREF1 short PREF2 adc100_open_unit (short port);
```

This routine opens the ADC-100 driver.

For DOS and the 16-bit Windows driver, it checks the BIOS printer address table and gets the address of the specified printer port. This is not possible in the Windows 32-bit driver, so it assumes that the printer ports 1..3 are at 0x378, 0x278 and 0x3BC.

It then calibrates the timing functions for the computer. It returns TRUE if successful. If it is not successful, you can call adc100_get_unit_info to find out why it failed.

port The number of the parallel port or USB port that the ADC-10 is connected to
- 1 - LPT1
- 2 - LPT2 etc
- 101 - USB-PP1
- 102 - USB-PP2 etc

adc100_close_unit

```
PREF 1 short PREF2 adc100_close_unit (short port);
```

This routine closes the ADC-100 driver.

port The number of the port

adc100_set_unit

```
PREF1 short PREF2 adc100_set_unit (short port);
```

This routine is used to select the unit to use for subsequent operations. It is only necessary to use this function if you wish to have more than one unit open at the same time.

adc100_set_range

```
PREF 1 void PREF2 adc100_set_range (short mv_a, short mv_b);
```

This routine sets the range for both channels. The two parameters are the input voltage ranges, in millivolts, for channels A and B. The ADC-100 is bipolar, so 20000 means that the input voltage range is ±20V.

Note that, for the ADC-101, the actual voltage range is always five times the voltage range specified. Thus, if mv_a is set to 20000, the actual voltage range is 100000, or 100V.

The following values will give the expected result: intermediate values will be rounded to the nearest above.
- 20000
- 10000
- 5000
- 2000
- 1000
- 500
- 200

100
…..50

If you are not using a channel, we recommend setting the range to ±20V. This prevents noise from the unconnected channel interfering with the channel that you are using.

adc100_get_value

```
PREF 1 short PREF2 adc100_get_value (short channel);
```

This routine reads the current value of one channel. Depending on your computer, it will take approx 20µs to take one reading.

`channel` 0 - channel A
1 - channel B

adc100_is_streaming

```
short adc100_is_streaming (void)
```

This routine can be used to determine whether the device is capable of supporting streaming. If so, it will return TRUE (1). A streaming device collects data asynchronously. USB devices generally support streaming, whereas parallel port devices do not.

adc100_run

```
void adc100_run (unsigned long no_of_values, unsigned short method)
```

This routine starts a streaming unit collecting data It collects readings at intervals and from channels specified in the most recent `adc100_set_interval` call. For non-streaming devices, this function has no effect.

`no_of_values` the number of samples to collect

`method` the data collection method:
BM_SINGLE (0) - collect a single block and stop
BM_WINDOW (1) - collect a sequence of overlapping blocks
BM_STREAM (2) - collect a continuous stream of data

adc100_ready

```
short adc100_ready (void)
```

This routine indicates whether a streaming device has completed its data collection. It returns TRUE if the device is ready to transfer data. For non-streaming devices, it always return TRUE.

adc100_stop

```
void adc100_stop (void)
```

This function cancels any pending request for data from a streaming device. It has no effect for non-streaming devices.

adc100_set_trigger

```
PREF1 void PREF2 adc100_set_trigger ( unsigned short enabled,
                                unsigned short auto_trigger,
                                unsigned short auto_ms,
                                unsigned short channel,
                                unsigned short dir,
                                unsigned short threshold,
                                unsigned short delay);
```

This routine defines a trigger event for the next block operation, and specifies the delay between the trigger event and the start of collecting the data block. Note that the delay can be negative for pre-trigger.

If the computer is stuck waiting for a trigger that never occurs, you can abort the data collection by pressing the F9 key (16-bit driver) or F10 (32-bit driver).

`enabled` this is `TRUE` if the ADC-100 is to wait for a trigger event, and `FALSE` if the ADC-100 is to start collecting data immediately.

`auto_trigger` this is `TRUE` if the ADC100 is to trigger after a specified time (even if no trigger event occurs). This prevents the computer from locking up, if no trigger event occurs.

`auto_ms` specifies the time in ms after which `auto_trigger` will occur.

`channel` specifies which channel is to be used as the trigger input.
0 - channel A
1 - channel B

`dir` the direction can be rising or falling.

`threshold` this is the threshold at which a trigger event on channel A or B takes place. It is scaled in ADC counts.

`delay` This specifies the delay, as a percentage of the block size, between the trigger event and the start of the block. Thus, 0% means the first data value in the block, and -50% means that the trigger event is in the middle of the block.

adc100_set_interval

```
PREF1 unsigned long PREF2 adc100_set_interval (
                                unsigned long us_for_block,
                                unsigned long ideal_no_of_samples,
                                short mode);
```

This routine specifies the time interval per sample and the channels to be used for calls to `adc100_get_values` or `adc100_get_times_and_values`.

`us_for_block` target total time in which to collect `ideal_no_of samples,` in micro seconds.

`ideal_no_of_samples` specifies the number of samples that you intend to collect. This number is only used for timing calculations: you can actually collect a different number of samples when you call `adc100_get_values`.

`mode` This is one of the following:
0 - channel A only

1 - channel B only
2 - both channels

`no_of_channels` specifies the number of channels used.

An example of a call to this routine using both channels A and B is:

adc100_set_interval (10000, 100, 2);

The routine returns the actual time to collect this number of samples. This actual time may be greater than the target time if you specified a sampling interval that is faster than your computer can manage. If the specified sampling rate was too fast, you have the following choices:

if the total time is important, collect fewer than the ideal number of samples so that the total block time is correct

if the number of samples is important, collect the same number of samples then allow for the fact that they took longer to collect.

adc100_get_values

```
PREF 1 unsigned long PREF2 adc100_get_values (
                                        unsigned short HUGE * buffer_a,
                                        unsigned short HUGE * buffer_b,
                                        unsigned long no_of_values);
```

This routine reads in a block of values. It collects readings at intervals and from channels specified in the most recent `adc100_set_interval` call.

If a key is pressed while collecting, the routine will return immediately. The return value will be zero if a key was pressed, and the total time in micro-seconds if a block was successfully collected.

When collecting data from just one channel, the parameter for the other buffer can either be set to NULL, or pointed at the same buffer.

adc100_get_times_and_values

```
PREF1 unsigned long PREF2 adc100_get_times_and_values (
                                    long HUGE * times,
                                    unsigned short HUGE * buffer_a,
                                    unsigned short HUGE * buffer_b,
                                    unsigned long no_of_values);
```

This routine reads a block of values from the unit in the most recent adc100_open_unit or adc100_set_unit call. It takes readings at nominal intervals specified in the most recent `adc100_set_interval` call, and returns the actual times for each reading.

If a key is pressed while collecting, the routine will return immediately. The return value will be zero if a key was pressed, and the total in micro-seconds if a block was successfully collected.

When collecting data from just one channel, the parameter for the other buffer can either be set to NULL, or pointed at the same buffer.

adc100_get_unit_info

```
PREF1 short PREF2 adc100_get_unit_info (char * str, short str_lth, short line, short
port);
```

If the specified unit failed to open, this routine returns a text string which explains why the unit was not opened.

If the specified unit is open, The routine returns version information about the ADC-100 DLL, the Windows driver and the sampling rate.

Str	- character string buffer for result
str_lth	- length of buffer
line	- 0 to 3: selects which line to return
port	- the printer port number (1..3) to return information for

adc100_get_combined_values

```
PREF1 unsigned long PREF2 adc100_get_combined_values
        (
        UNS16                   channel,
        COMBINATION_METHOD      mode,   /* Combination modes (CM_XXX) */
        UNS16                   no_of_readings  )
```

This routine takes a set of readings from the specified channel, at full speed, and returns either the mimimum, maximum, average or sum of the set of readings.

Channel	0 – channel A 1 – channel B
mode	0 – average 1 – minimum 2 – maximum 3 – sum
no_of_readings	the number of readings to take

DOS

From DOS, it is possible to access the ADC-100 in C and Pascal using the DOS driver. It is not possible to call the ADC-100 DOS driver from BASIC.

The driver uses PASCAL linkage conventions.

The DOS driver does not support huge memory, so the data buffer must be less than 64k bytes.

See ADC100.txt for more information about the DOS driver.

Windows 3.x

In Windows 3.1 it is possible to use the 16-bit Windows driver, or to access the ADC-100 directly.

When running under Windows 3.x, an application is not in complete control- Windows can interrupt at any time. Interruptions occur every 55 milliseconds, and are also caused by mouse and keyboard input. As a consequence, the driver cannot always take readings at fixed time intervals. To deal with this, the driver returns the time at which each reading was taken.

The Windows 16-bit driver is called PICO.386, and is installed in windows\system. It is loaded using a reference in system.ini:

[386enh]
.....
.....
device=pico.386

The driver is accessed using the file `ADC10016.DLL`: this is installed in the drivers\win sub-directory: for some applications (eg Visual Basic), it is necessary to copy the DLL to `c:\windows\system`.

The DLL uses PASCAL linkage conventions, and uses HUGE pointers to data items, so that C and Delphi programs can access arrays larger than 64k bytes.

Examples are provided for C, Delphi, Visual Basic and Excel.

Windows 95/98/ME

In Windows 95, 98 and ME, you can use the 16-bit or the 32-bit driver: it is also possible to access the ADC-100 directly.

The following applications require 32-bit driver:

- Visual Basic 4 and above
- Excel 7 and above
- Delphi 2 and above
- Borland C 5
- Microsoft C version 2 and above.
- LabVIEW version 4 and above

The 16-bit and 32-bit drivers do not interfere with each other, so it is possible to install both drivers on the same system, as long as, for any given unit, only one driver is using it at once.

When running under Windows 95, an application is not in complete control- Windows can interrupt at any time. Interruptions occur every 55 milliseconds, and are also caused by mouse and keyboard input. As a consequence, the printer port driver cannot always take readings at fixed time intervals. To deal with this, the driver returns the time at which each reading was taken. Generally speaking, the 16-bit driver gives higher sampling rates, but the 32-bit driver is less prone to large gaps in the data.

The Windows 95 32-bit driver, PICO.VXD, is installed in windows\system, It is loaded using a reference in system.ini:

[386enh]
.....
.....
device=pico.VXD

The Windows 98/ME USB port driver, PICOPP.SYS, is installed in \windows\system32\drivers. The file picopp.inf, must be placed in \windows\inf so that Windows knows which driver to load when the USB parallel port is plugged in.

The Windows 95 32-bit driver is accessed using the file `ADC10032.DLL`: it is installed in `drivers\win32`. The DLL uses STDCALL linkage conventions, and undecorated names. The same `ADC10032.dll` file can be used in all 32-bit versions of Windows, for both parallel port and USB port connected products.

Windows NT/2000/XP

The Windows NT/2000/XP driver, PICO.SYS, is installed in windows\system32\drivers. The operating system must be told that the driver is available: this is normally done automatically by the setup program, but can also be done manually using the the regdrive.exe program which is copied into the PICO directory. Type in

```
regdrive pico
```

The Windows 2000/XP USB port driver, PICOPP.SYS, is installed in \windows\system32\drivers. The file picopp.inf, must be placed in \windows\inf so that Windows knows which driver to load when the USB parallel port is plugged in.

The Windows NT 32-bit driver is accessed using the file `ADC10032.DLL`: it is installed in `drivers\win32`. The DLL uses STDCALL linkage conventions, and undecorated names. The same `ADC10032.dll` file can be used in all 32-bit versions of Windows, for both parallel port and USB port connected products.

C

DOS
To link the driver into you program, you should take the following steps:

`#include` the header file `adc100.h` into your program
If you are using an IDE, include the file `adc100drv.obj` in you project.
If you are using a command-line compiler, include the file `adc100drv.obj` in you linkfile.

See `adc100b.c` for an example of a simple DOS program which uses the driver.

C / C++ (Windows)

C

There are two C example programs: one is a very simple GUI application, and the other is a more comprehensive console mode program that demonstrates all of the facilities of the driver.

The GUI example program is a generic windows application- ie it does not use Borland AppExpert or Microsoft AppWizard. To compile the program, create a new project for a Windows Application containing the following

files:
a100test.c
a100test.rc
either adc10016.lib (All 16-bit applications)
or adc10032.lib (Borland 32-bit applications)
or adc100ms.lib (Microsoft Visual C 32-bit applications)

The following files must be in the same directory:
a100test.rch
adc100.h
either adc10016.dll (All 16-bit applications)
or adc10032.dll (All 32-bit applications)

The console example program is a generic windows application- ie it does not use Borland AppExpert or Microsoft AppWizard. To compile the program, create a new project for a Console Application containing the following files:
a100con.c
either adc10016.lib (All 16-bit applications)
or adc10032.lib (Borland 32-bit applications)
or adc100ms.lib (Microsoft Visual C 32-bit applications)

The following files must be in the same directory:
adc100.h
either adc10016.dll (All 16-bit applications)
or adc10032.dll (All 32-bit applications)

C++

C++ programs can access all versions of the driver. If adc100.h or adc100w.h are included in a C++ program, the PREF1 macro expands to **extern "C"**: this disables name-mangling (or decoration, as Microsoft call it), and enables C++ routines to make calls to the driver routines using C headers.

Pascal

The program `adc100.pas` can be compiled either as a stand-alone program `{$DEFINE MAIN}` or as a unit which can be linked into other programs `{$UNDEF MAIN}`.

`adc100.pas` includes the driver using the `{$L adc100drv.obj}` command: it also provides pascal prototypes for each of the routine in the driver.

This program has been tested with Borland Turbo Pascal V6.0.

Basic

The DOS driver does not work with DOS Basic.

Delphi

`adc100pr.dpr` is a complete program which opens the driver and reads values from channel 1.

The file `ADC100.inc` contains a set of procedure prototypes that you can include into your own programs.

Excel

The easiest way to get data into Excel is to use the Picolog for Windows program.

However, you can also write an Excel macro which calls `adc100xx.dll` to read in a set of data values. The Excel Macro language is similar to Visual Basic.

The example `ADC100xx.XLS` reads in 20 values from channels 1 and 2, one per second, and assigns them to cells A1..B20.

Use 16-bit driver for Excel version 5, and the 32-bit driver for Excel version 7 and above.

Note that it is usually necessary to copy the .DLL file to your \windows\system directory.

Visual Basic

Version 3 (16 bits)
The DRIVERS\WIN16 sub-directory contains a simple Visual Basic program, `ADC100.mak`.

ADC10016.MAK
ADC10016.FRM

Note that it is usually necessary to copy the .DLL file to your \windows\system directory.

Version 4 and 5 (32 bits)
The DRIVERS\WIN32 sub-directory contains the following files:

ADC10032.VBP
ADC10032.BAS
ADC10032.FRM

LabVIEW

The routines described here were tested using LabVIEW for Windows 95 version 4.0.

While it is possible to access all of the driver routines described earlier, it is easier to use the special LabVIEW access routines if only single readings are required. The adc100.llb library in the `DRIVERS\WIN32` sub-directory shows how to access these routines.

To use these routines, copy `adc100.llb` and adc10032.dll to your LabVIEW user.lib directory. You will then find three sub-vis to access the ADC-100 and ADC-101.

Adc100_single takes a single reading from a specified port and channel
adc100_example shows how to call adc100_single repeatedly.
Adc100_block shows how to collect a block of data at high speeds.

HP-Vee

The example program adc100.vee is in the drivers\win32 sub-directory. It was tested using HP-Vee version 5 under Windows 95.

The example shows how to collect a block of data from the adc-100. It would be necessary to adjust the scaling for use with the ADC-101.

Linux

The ADC100 and ADC101 are supported under Linux using the `picopar` parallel port driver kit. The tar

file `picopar.tar`, available from the Pico web site, contains source code for the driver and example programs, together with full instructions to compile, install and run the software.

The linux parallel port driver kit supports only units connected direct to the parallel port: it does not support USB-connected devices.

Update history

28Feb02 MKG Added description of USB and Linux support
15May02 MPB Updated safety warning for LVD compliance
29May02 MPB Updated USB information following review
31May02 MPB Reworded USB parallel port operation